Branch: master ▾ Find file Copy path

7331DataMiningNotebooks / lab1 / LAB01-DataMining-Joe Copy.ipynb

joseph.schueder@collins.com add pca from daniel

0fe1f14 1 hour ago

2 contributors

Download History

2.12 MB

Assignment 1

Discussion Notes:

- Whiskey vs Non-Whiskey (Classification) to be used
- 400K dataset to be used

Next steps:

- Clean up business understanding
- jeff to put images of barplots of entire dataset within exceptional work
- review/edit write up regarding classification within business rule
- run and clean up code within each section
- add any necessary verbiage for each image

Todo:

- provide a bar chart showing the months represented in the dataset (in visualizations). can include something like sales in the month
- make sure we are answering each quesiton in the rubric in the respective section, so move our outliers discussion and how we addressed them into the data quality section
- provide a few additional visualizations on whiskies
- possibly get a tableau visual of iowa zip codes and whiskies
- do an LDA on whiskey in the exceptional section (dan can work on)

Business Understanding

Introduction - The Iowa Liquor Sales dataset is an API from Google's Bigquery which contains the wholesale purchases by retail stores in the Iowa area. The dataset includes the spirit purchase details by product, date of purchase, and location the item was purchased from an Iowa Class "E" liquor license holder (retail stores) . The timeframe of this data starts from January 1, 2012 through 2019. As part of the study commissioned by the Iowa Department of Commerce, all alcoholic sales within the state were logged into the Department system, and in turn, published as open data by the State of Iowa.

Load Python Packages

In [5]:
```
# Import necessary python packages
try:
    from collections import abc as collections_abc
except ImportError:  # Python 2.7
    import collections as collections_abc

    import copy
import functools
import gzip
import io
import itertools
import json
import math
import os
import tempfile
import uuid
import warnings
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. p
d.read_csv)
import pandas as pd
import numpy as np
#import altair as alt
import matplotlib.pyplot as plt
import re

# Imports the Google Cloud client library
#from google.cloud import storage
from google.oauth2 import service_account
from google.cloud import bigquery
```

Load Data from github

for the year

For our measurement, we will be querying the 2019 and limiting the rows to 50,000 for our intial modeling, which we can later expand to see how our model translates to scale.

In [6]:
```
# Read csv from disk
```



```
df =  pd.read_csv(r'/Users/jjschued/Documents/Github/7331Da
taMiningNotebooks/lab1/iowa_subset_2019_400k_random_rows.cs
v', nrows = 100000)
#df =  pd.read_csv(r'/Users/danielclark/Desktop/SMU/data_mi
ning/7331DataMiningNotebooks/lab1/iowa_subset_2019_400k_ran
dom_rows.csv', nrows = 50000)

# read csv from github directly
#url = 'https://github.com/jjschueder/7331DataMiningNoteboo
ks/blob/master/lab1/iowa_subset_2019_400k_random_rows.csv?r
aw=true'
#df = pd.read_csv(url, nrows=50000)

# verify data read in
print(df.head(5))
```

```
  invoice_and_item_number        date  store_number  \
0         INV-23084400002  2019-11-07          3869
1         INV-23282200001  2019-11-18          4617
2         INV-19435800130  2019-05-17          2560
3         INV-23454200002  2019-11-25          2614
4         INV-23541200001  2019-11-27          2629

                              store_name
address  \
0               Bootleggin' Barzini's Fin                41
2  1st Ave
1                          Lickety Liquor            2501 H
UBBELL AVE
2              Hy-Vee Food Store / Marion  3600 Business Hw
y 151 East
3  Hy-Vee #3 Food & Drugstore / Davenport          1823 E K
imberly Rd
4   Hy-Vee Food Store #2 / Council Bluffs            1745 M
adison Ave

            city  zip_code                         store_lo
cation  \
0      Coralville   52241.0          POINT (-91.565517 41.6
72672)
1      Des Moines   50317.0          POINT (-93.570489 41.6
07817)
2         Marion   52302.0
NaN
```

```
99999      141.00          12.00          3.1
7

[100000 rows x 24 columns]>
```

Running a df.columns.values function, confirms the 24 features that we referenced in our Data understanding phase. This now allows us to move forward with our data cleaning.

```
In [38]: df.columns.values

Out[38]: array(['invoice_and_item_number', 'date', 'store_number',
       'store_name',
       'address', 'city', 'zip_code', 'store_location', 'co
unty_number',
       'county', 'category', 'category_name', 'vendor_numbe
r',
       'vendor_name', 'item_number', 'item_description', 'p
ack',
       'bottle_volume_ml', 'state_bottle_cost', 'state_bott
le_retail',
       'bottles_sold', 'sale_dollars', 'volume_sold_liter
s',
       'volume_sold_gallons'], dtype=object)
```

move box plots here. for outlier detection

Data Cleaning

In our data cleaning step, we need to perform a few specific data cleaning operations to:

1. Convert our features to the correct continuous, ordinal and categorical features

   Replace the values for pack, bottle_volume_ml, store_number, store_name, address, city, zip_code, county_number, county, category, category_name, vendor_number, vendor_name, item_number and item_description into categorical variables so they register as a non-null object in our model

2. Address the missing values

   Replace all "?", which our dataset denotes as null, into "-1" values (not strings). From here, we will convert state_bottle_cost, state_bottle_retail, sale_dollars, volume_sold_liters, and volume_sold_gallons into continuous variables so they register as floats.

3. Create a category that simplifies our alcohol categories to specific genres like whiskey, vodka, tequilla, ect.
4. Categorize our store locations into a few easily discernable buckets
5. Create a month and date column for opportunities to time and date analysis
6. Convert the varaible "bottles_sold" into ordinal features so they register as an integer value in our models

no duplicates?

Using a df.types function helps to verify this.

```
In [39]: df.dtypes

Out[39]: invoice_and_item_number     object
         date                        object
         store_number                 int64
         store_name                  object
         address                     object
         city                        object
         zip_code                   float64
         store_location              object
         county_number              float64
         county                      object
         category                   float64
         category_name               object
         vendor_number                int64
         vendor_name                 object
         item_number                  int64
         item_description            object
         pack                         int64
         bottle_volume_ml             int64
         state_bottle_cost          float64
         state_bottle_retail        float64
         bottles_sold                 int64
         sale_dollars               float64
         volume_sold_liters         float64
         volume_sold_gallons        float64
         dtype: object
```

```
df.loc[df['month'] ==11 , 'month'] = 'Nov'
df.loc[df['month'] ==12 , 'month'] = 'Dec'


df.loc[df['year'] ==2012 , 'year'] = '2012'
df.loc[df['year'] ==2013 , 'year'] = '2013'
df.loc[df['year'] ==2014 , 'year'] = '2014'
df.loc[df['year'] ==2015 , 'year'] = '2015'
df.loc[df['year'] ==2016 , 'year'] = '2016'
df.loc[df['year'] ==2017 , 'year'] = '2017'
df.loc[df['year'] ==2018 , 'year'] = '2018'
df.loc[df['year'] ==2019 , 'year'] = '2019'

#merge year and month together
df['monthyear'] = df['month'] + "-" + df['year']
```

In [59]:
```
df.head(20)
```

Out[59]:

	invoice_and_item_number	date	store_number	store_name
0	INV-23084400002	2019-11-07	3869	BOOTLEGGIN' BARZINI'S FIN
1	INV-23282200001	2019-11-18	4617	LICKETY LIQUOR
2	INV-19435800130	2019-05-17	2560	HY-VEE FOOD STORE / MARION
3	INV-23454200002	2019-11-25	2614	HY-VEE #3 FOOD & DRUGSTORE / DAVENPORT
4	INV-23541200001	2019-11-27	2629	HY-VEE FOOD STORE #2 / COUNCIL BLUFFS
		2019-		I-80 LIQUOR /

5	INV-23540300028	2019-11-27	4312	COUNCIL BLUFFS
6	INV-19655400089	2019-05-29	2517	HY-VEE FOOD STORE #1 / NEWTON
7	INV-23687700099	2019-12-05	2545	HY-VEE DRUGSTORE / IOWA CITY
8	INV-17141800065	2019-01-25	5210	DING'S HONK'N HOLLER
9	INV-23109300014	2019-11-08	5411	BLUEJAY MARKET
10	INV-23072700074	2019-11-06	2642	HY-VEE WINE AND SPIRITS / PELLA
11	INV-19202100034	2019-05-07	5173	AUDUBON FOOD LAND
12	INV-23553600002	2019-11-29	2190	CENTRAL CITY LIQUOR, INC.
13	INV-23551100140	2019-11-27	2565	HY-VEE FOOD STORE #1636 / SPENCER
14	INV-23566000005	2019-11-29	4192	FAREWAY STORES #044 / BETTENDORF
15	INV-19970300001	2019-06-13	2675	HY-VEE #2 / CORALVILLE
16	INV-20083400001	2019-	3773	BENZ

```
Boolean Columns:  0
Nominal Columns:  21
Continuous Columns:  8
Columns Accounted for:  29
```

add text about this

In [20]:
```
#Delete categorical columns with > 30 unique values (Each u
nique value becomes a column during one-hot encoding)
oneHotUniqueValueCounts = df[D_nominal.columns].apply(lambd
a x: x.nunique())
oneHotUniqueValueCols = oneHotUniqueValueCounts[oneHotUniqu
eValueCounts >= uniqueThreshold].index
```

In [21]:
```
oneHotUniqueValueCounts
oneHotUniqueValueCols
```

Out[21]:
```
Index(['invoice_and_item_number', 'date', 'store_number',
'store_name',
       'address', 'city', 'zip_code', 'store_location', 'co
unty_number',
       'county', 'category', 'category_name', 'vendor_numbe
r', 'vendor_name',
       'item_number', 'item_description'],
      dtype='object')
```

In [22]:
```
oneHotUniqueValueCols
```

Out[22]:
```
Index(['invoice_and_item_number', 'date', 'store_number',
'store_name',
       'address', 'city', 'zip_code', 'store_location', 'co
unty_number',
       'county', 'category', 'category_name', 'vendor_numbe
r', 'vendor_name',
       'item_number', 'item_description'],
      dtype='object')
```

In [23]:
```
onehotlist = oneHotUniqueValueCols.tolist()
onehotlist
```

Out[23]:
```
['invoice_and_item_number',
 'date',
 'store_number',
 'store_name',
 'address',
 'city',
 'zip_code',
 'store_location',
 'county_number',
 'county',
 'category',
 'category_name',
 'vendor_number',
 'vendor_name',
 'item_number',
 'item_description']
```

In [24]:
```
#one hot encoding
dfenc = df.copy()
dfenc.head()
```

make this small

Out[24]:

	invoice_and_item_number	date	store_number	store_name
0	INV-23084400002	2019-11-07	3869	BOOTLEGG BARZINI'S F
1	INV-23282200001	2019-11-18	4617	LICKETY LIQUOR
2	INV-19435800130	2019-05-17	2560	HY-VEE FO STORE / MARION
3	INV-23454200002	2019-11-25	2614	HY-VEE #3 FOOD & DRUGSTOR DAVENPOR
4	INV-23541200001	2019-11-27	2629	HY-VEE FO STORE #2 / COUNCIL BLUFFS
5	INV-23540300028	2019-11-27	4312	I-80 LIQUOF COUNCIL BLUFFS

99996	INV-22567100020	2019-10-15	5351	DOWNTOW LIQUOR
99997	INV-22471800013	2019-10-11	2607	HY-VEE WIN AND SPIRIT SHENANDC
99998	INV-22549300142	2019-10-15	2651	HY-VEE / WAVERLY
99999	INV-22741000016	2019-10-23	3773	BENZ DISTRIBUTI

100000 rows × 29 columns

In [29]:
```
#'month', 'year', 'county_number',
#Isolate continuous and categorical data types
#These are indexers into the schoolData dataframe and may b
e used similar to the schoolData dataframe
dfenc_boolean = dfenc.loc[:, (dfenc.dtypes == bool) ]
dfenc_nominal = dfenc.loc[:, (dfenc.dtypes == object)]
dfenc_continuous = dfenc.loc[:, (dfenc.dtypes != bool) & (d
f.dtypes != object)]
print ("Boolean Columns: ", dfenc_boolean.shape[1])
print ("Nominal Columns: ", dfenc_nominal.shape[1])
print ("Continuous Columns: ", dfenc_continuous.shape[1])
print ("Columns Accounted for: ", dfenc_nominal.shape[1] +
dfenc_continuous.shape[1] + dfenc_boolean.shape[1])

one_hot_df = pd.concat([pd.get_dummies(dfenc[col],prefix=co
l) for col in dfenc_nominal.columns], axis=1)
one_hot_df.head()
```

```
Boolean Columns:  0
Nominal Columns:  5
Continuous Columns:  8
Columns Accounted for:  13
```

Out[29]:

	liquor_category_AMARETTO	liquor_category_BRANDY	liquor_ca
0	0	0	0

1	0	0	0
2	0	0	1
3	0	0	0
4	0	0	0

5 rows × 46 columns

In [30]:
```
df1hotmerge = pd.merge(dfenc, one_hot_df, left_index=True,
right_index=True)
df1hotmerge
```
+ head()

df1hotmerge.info() to check again for nulls

- do we move one hot to exceptional work since we aren't using it later?

Out[30]:

	pack	bottle_volume_ml	state_bottle_cost	state_bottle_retai
0	20	375	3.85	5.78
1	8	50	8.75	13.13
2	12	1000	16.50	24.75
3	6	750	21.17	31.76
4	6	1750	9.31	13.97
5	12	1000	28.98	43.47
6	12	500	4.90	7.35
7	6	750	8.50	12.75
8	8	300	8.75	13.13
9	6	1000	33.34	50.01
10	12	750	4.47	6.71
11	12	1000	3.00	4.50
12	10	600	5.95	8.93
13	12	1000	3.01	4.52
14	12	750	12.49	18.74

On the one hot encoding dataset, we are going to drop the columns, vendor_number, item_number, and store_number as they will be redundant to other features in our dataset.

From here, lets see how big our dataset gets when we onehot our categorical variables.

In [31]:
```
df[df.isnull().any(axis=1)][df.columns[df.isnull().any()]]
```

Out[31]: □

In [32]:
```
# this python magics will allow plot to be embedded into the notebook
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.simplefilter('ignore', DeprecationWarning)
%matplotlib inline
```

move this to simple statistics

In [33]:
```
df['liquor_category'].value_counts()
```

Out[33]:
```
VODKA         26316
WHISKY        22645
RUM           16546
LIQUEUR       15872
TEQUILA        6916
Other          6568
GIN            5067
AMARETTO         45
BRANDY           19
SCHNAPPS          6
Name: liquor_category, dtype: int64
```

Looking at a quick count of our categories, we can see that Vodka appears the most often in our dataet, at over 13,000 times followed by Whisky with over 10,000 instances.

Below, we created a column called cost_per_liter and isolated the sales that exceeded $20,000. There were 7 total values with 6 of the 7 being Vodka Varieties. However, note the cost of liter is not that high in relation to to the sale and the volume sold per liters are almost all in the quadruple digits, which suggest these are cheaper alcohols being sold in high volume. 4 of the 7 sales were at Hy-Vee, which we will look at soon.

In [34]:
```
# selecting rows based on condition
exp_df = df[df['sale_dollars'] > 20000]

exp_df['cost_per_liter'] = exp_df['sale_dollars']/exp_df['volume_sold_liters']

print(exp_df[['sale_dollars', 'volume_sold_liters', 'cost_per_liter', 'liquor_category', 'store_parent']])
```

outlier analysis move up?

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
      sale_dollars  volume_sold_liters  cost_per_liter liquor_category  \
1480       25009.92              2016.0       12.405714         VODKA
12772      22588.80              1170.0       19.306667           RUM
24758      31347.00              4725.0        6.634286         VODKA
25507      22668.00              1200.0       18.890000         VODKA
32191      25239.36              1098.0       22.986667         VODKA
32454      20688.00               900.0       22.986667         VODKA
42212      37514.88              3024.0       12.405714         VODKA
61110      24062.40              2520.0        9.548571       LIQUEUR
63553      30856.32              1056.0       29.220000       WHISKY
66805      31347.00              4725.0        6.634286         VODKA
67216      34481.70              5197.5        6.634286         VODKA
```

```
dtypes: float64(6), int64(2), object(21)
memory usage: 22.1+ MB
```

In [36]:
```
df.describe()
```

Out[36]:

	pack	bottle_volume_ml	state_bottle_cost	state_b
count	100000.000000	100000.000000	100000.000000	100000.
mean	12.206910	872.891780	12.764987	19.1500
std	8.824624	465.055798	11.231683	16.8474
min	1.000000	50.000000	0.890000	1.34000
25%	6.000000	750.000000	4.990000	7.49000
50%	12.000000	750.000000	10.450000	15.6800
75%	12.000000	1000.000000	17.160000	25.7675
max	48.000000	6000.000000	262.460000	393.690

In [37]:
```
dfenc.mode()
```

Out[37]:

	pack	bottle_volume_ml	state_bottle_cost	state_bottle_retail	bo
0	12	750	18.09	27.14	12.

Some Interesting facts can be found in the mode section. Vodka has the largest number of sales, and the small package stores combined take up more sales than a single big-box retailer.

The most common pack size is 12 pack, which is efficient for shipping and the most common bottle size is 750 ml.

December 2019 is the most popular month for alcohol sales, which suggest that people drink heavily that time of year. :)

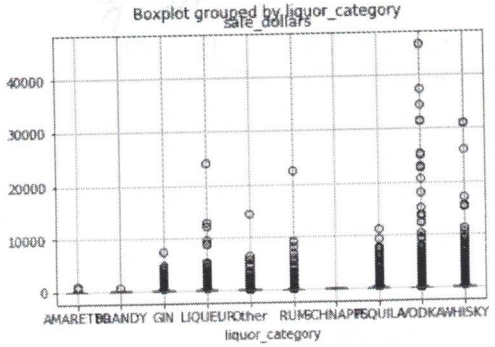
Visualize Attributes

Visualize the most interesting attributes (at least 5 attributes, your opinion on what is interesting). Important: Interpret the implications for each visualization. Explain for each attribute why the chosen visualization is appropriate.

The following section shows our cross tabulations of the relationships we found with our 2019 liquor data. The bar charts and box plots are ablet o show the distribution of our data and a comparative between categorical variables in our set.

In [38]:
```
# pandas has some really powerful extensions to matplotlib
 for scientific computing
ax = df.boxplot(column = 'sale_dollars', by = 'liquor_categ
ory')
ax
```

Out[38]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x29db2d9e828>
```



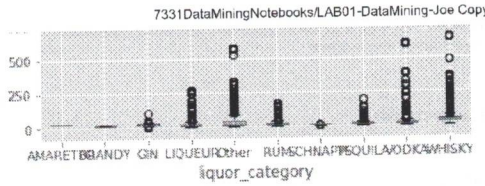
— move up to outlier

This box plot is graphing the distribution sale_dollars with the different categories of liquor we have available. Here, we can see there is a significant number of outliers across all categories, so much that you can't really see the boxes on each category. This would suggest we would want to look into a transform.

In [39]:
```
df['sale_dollars_trans'] = np.log(df['sale_dollars'])
```

4	INV-23541200001	2019-11-27	2629	Food Store #2 / Council Bluffs
5	INV-23540300028	2019-11-27	4312	I-80 Liquor / Council Bluffs
6	INV-19655400089	2019-05-29	2517	Hy-Vee Food Store #1 / Newton
7	INV-23687700099	2019-12-05	2545	Hy-Vee Drugstore / Iowa City
14	INV-23566000005	2019-11-29	4192	Fareway Stores #044 / Bettendorf
15	INV-19970300001	2019-06-13	2675	Hy-Vee #2 / Coralville
24	INV-23645000007	2019-12-03	5866	Hometown Family Market
25	INV-19684100047	2019-05-30	4073	Uptown Liquor, Llc
29	INV-19201600066	2019-05-07	2651	Hy-Vee / Waverly
33	INV-16789600092	2019-01-07	2573	Hy-Vee Food Store / Muscatine
34	INV-20689800014	2019-07-17	5293	The Boonedocks
36	INV-23871500052	2019-12-12	4988	Happy's Wine & Spirits

37	INV-23844500030	2019-12-11	5293	The Boonedocks
39	INV-17028600022	2019-01-18	4522	Casey's General Store #2523 / Monroe
43	INV-20141300151	2019-06-21	2560	Hy-Vee Food Store / Marion
44	INV-20841100115	2019-07-25	4129	Cyclone Liquors
46	INV-23686300014	2019-12-05	2675	Hy-Vee #2 / Coralville
56	INV-23794300008	2019-12-10	5592	Hubers Store
60	INV-23300600126	2019-11-18	4167	Iowa Street Market, Inc.
206	INV-19688800021	2019-05-30	4698	Quality Quick Stop
243	INV-23556700122	2019-11-29	4129	Cyclone Liquors
1011	INV-22289800001	2019-10-03	4677	Costco Wholesale #1111 / Coralville
		2019-		Costco Wholesale



→ more up

Looking at the cost per liter aggregate column. As we discussed previously, cost per liter is calculated via a calculation of total sales_dollars divided by the total volume_sold_ml. The idea of this calculation is to see the varying price of the liquors if we normalize by the volume and sale of the liquors in our set.

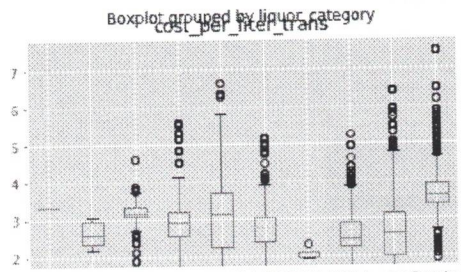
Here, we can see that our distribution is right skewed with a large outlier for whiskey which is going for over 1750 per liter. Looking further into this datapoint, we can see that this datapoint represents Johnnie Walker Blue.

That said, with the distribution of alcohol types, outside of Johnnie walker blue, we cans see that our cost per liter is going to have a greater than tenfold range, so, i would suggest we create a transform there as well.

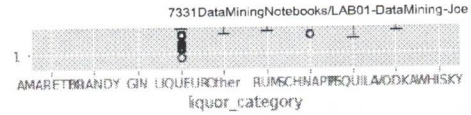
In [44]:
```
df['cost_per_liter_trans'] = np.log(df['cost_per_liter'])

zx = df.boxplot(column = 'cost_per_liter_trans', by = 'liqu
or_category')
zx
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x29db8811da0>



← more up

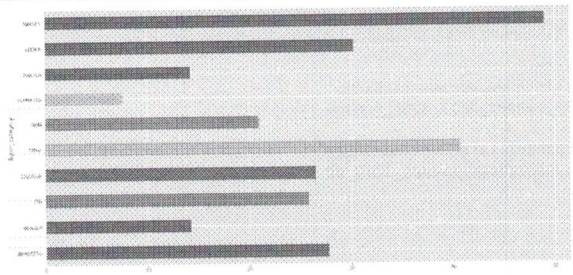


This new box plot shows some interesting insight into the distribution and the value placed on alcohol in our dataset.

Gin and Tequilla - even with outliers, it is going to have a faily consistent price per liter. Liquer - Similar case as Gln, however, it's outliers go a little farther out. Other- as expected, has a very wide range with very few outliers. Since this represents all types of alcohol not covered in the other categories, the box plot shape is expected.

Whiskey - Has a faily consistent bounding, however the number out outliers on either side is very large and larger than the other categories.

In [45]:
```
# Start by just plotting what we previsously grouped!
plt.style.use('ggplot')
plt.figure(figsize=(20,10))
df_grouped = df.groupby(by=['liquor_category'])
sales_rate = df_grouped.cost_per_liter.mean()
ax = sales_rate.plot(kind='barh')
```



— group !

This plot shows the grouped average cost per liter by the type of liquor in our dataset. Coming in at over 50 dollars per liter, whiskey is the most expensive alcohol in our dataset. However, some of that can come from the Johnnie Walker Blue we discussed previously, the common theme is that whiskey tends to be more expensive liter for liter

Edrington Group USA LLC	185043.45
MOET HENNESSY USA	168206.25
CEDAR RIDGE VINEYARDS LL	142926.26

As a function of bottles sold, we are seeing a similar distirbution with regards to sale_dollars in our top whiskies. Pernod Richard is nearing 3 million dollars in sales followed by Diageo Americas that is under 1 million dollars in sales.

In [71]:
```
dfscpl.sort_values(by='cost_per_liter', ascending=False).he
ad(10)
```

Out[71]:

	cost_per_liter
vendor_name	
PACIFIC EDGE WINE & SPIRITS	145.564444
IMPEX BEVERAGE INC	123.086667
HOTALING & CO	104.243810
W J Deutsch & Sons LTD	93.320000
GoAmericaGo Beverages LLC	89.461075
MOET HENNESSY USA	74.180870
Foundry Distilling Company, LLC	71.983333
Vision Wine & Spirit LLC	69.010256
Paterno Imports LTD	68.098667
MISSISSIPPI RIVER DISTIL	66.810435

Interestingly, looking at the high cost per liter vendors, Pactific Edge Wine and Spirits has the highest cost per liter at over 115 dollars, followed by Impex Beverage and Hotaling at just over 100 dollars.

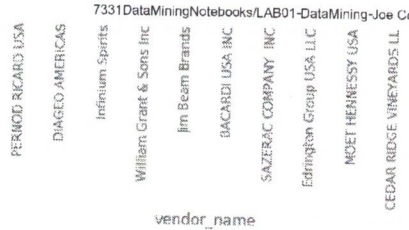
In [72]:
```
salesbyvendor = pd.merge(dfsr, dfsdol, how = 'left', on='ve
```



```
ndor_name')
salesbyvendor = pd.merge(salesbyvendor, dfscpl, how = 'lef
t', on='vendor_name')
salesbyvendor = pd.merge(salesbyvendor, dfsg, how = 'left',
on='vendor_name')
salesbyvendor
```

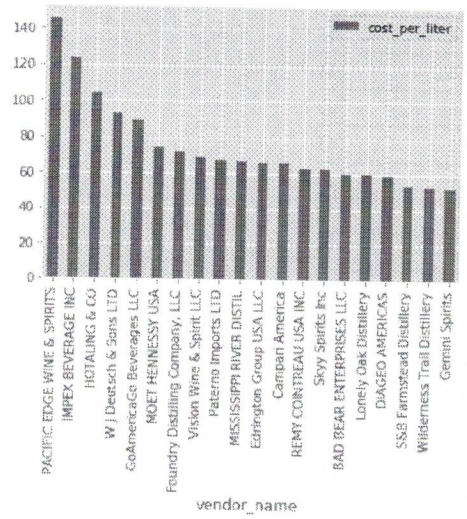
Out[72]:

	bottles_sold	sale_dollars	cost_per_liter	grossm
vendor_name				
American Heritage Distillers, LLC	12	189.60	21.066667	0.33354
BACARDI USA INC	7716	216439.87	28.791171	0.33333
BAD BEAR ENTERPRISES LLC	413	18987.00	60.047619	0.33333
Brown Forman Corp.	2284	68098.15	41.052515	0.33343
CEDAR RIDGE VINEYARDS LL	4411	142926.26	44.172657	0.33339
CHATHAM IMPORTS INC	345	11489.06	48.234476	0.33341
CONSTELLATION BRANDS INC	468	17241.75	49.780488	0.33333
COOPER SPIRITS INTERNATIONAL	120	450.00	37.500000	0.33333
Campari America	268	11031.00	66.564706	0.33333
Castle Brands	232	3792.74	21.614884	0.33355
Cats Eye Distillery	199	5751.56	38.514497	0.33344



In [77]:
```
# Start by just plotting what we previsously grouped!
plt.style.use('ggplot')
plt.figure(figsize=(20,10))
sales_rate = dfscpl.sort_values(by='cost_per_liter', ascend
ing=False).head(20)
ax = sales_rate.plot(kind='bar')
```

```
<Figure size 1440x720 with 0 Axes>
```



Same as tables format together

Above are some additional plots to help you visualize a more complete dataset on cost per liter as well as sales dollar of our specific whiskeis in the study. As discussed before, the Pernod Richard whiskey is the top selling vendor,while Pacific Edge Whiskey has the highest cost per liter, ensinuating they are seeling the most valueable whiskey.

mage.png

If we are interested in big box retailers, lets look into Hy-Vee.

In [78]:
```
hv_df = pd.get_dummies(df['store_parent'], drop_first=False
)

hv_df = pd.concat([df, hv_df], axis = 1, sort=False)
```

In [79]:
```
hv_df.head()
```

Out[79]:

	invoice_and_item_number	date	store_number	store_name	a
0	INV-23084400002	2019-11-07	3869	BOOTLEGGIN' BARZINI'S FIN	
1	INV-23282200001	2019-11-18	4617	LICKETY LIQUOR	
2	INV-19435800130	2019-05-17	2560	HY-VEE FOOD STORE / MARION	
3	INV-23454200002	2019-11-25	2614	HY-VEE #3 FOOD & DRUGSTORE / DAVENPORT	
4	INV-23541200001	2019-11-27	2629	HY-VEE FOOD STORE #2 / COUNCIL BLUFFS	

	Hy-Vee	6.000000	750.000000	13.86(
AMARETTO	Other	6.000000	750.000000	13.86(
	SamsClub	6.000000	750.000000	13.86(
	Target	6.000000	750.000000	13.86(
BRANDY	Hy-Vee	12.000000	791.666667	6.011(
	Other	8.076923	1000.000000	9.640(
GIN	CVS	6.000000	1562.500000	24.00(
	Caseys	23.684211	388.157895	7.095:
	Hy-Vee	9.901179	986.244408	16.23'
	Other	10.393676	983.467003	16.32(
	QuikTrip	12.000000	1000.000000	15.73(
	SamsClub	7.987013	1287.337662	20.35(
	SmokingJoes	20.727273	477.272727	6.853(
	Target	6.517241	1077.586207	20.88:
	Wal-Mart	7.793814	1391.752577	20.78'
	Walgreens	24.000000	375.000000	6.990(
LIQUEUR	CVS	11.400000	1012.500000	12.91:
	Caseys	23.378747	463.555858	6.603:
	Hy-Vee	12.550103	856.168733	9.823(
	Kum&Go	22.610837	432.142857	5.894(
	Other	14.269837	801.338762	9.741(
	QuikTrip	31.406061	413.939394	5.080:
	SamsClub	11.273159	1103.503563	11.18:
	SmokingJoes	20.681648	446.067416	6.531:
	Target	7.959596	1879.292929	25.02:
	Wal-Mart	10.700370	1075.709001	12.23:

	Walgreens	14.956522	705.434783	10.98'
Other	Caseys	8.521739	1050.000000	7.753(
	Hy-Vee	8.638420	872.450153	13.72(
	Kum&Go	11.333333	638.888889	5.018(
...	...	...	...	...
TEQUILA	Kum&Go	9.428571	1035.714286	7.951(
	Other	14.993648	1065.093769	9.460'
	QuikTrip	40.982143	365.625000	4.696'
	SamsClub	10.701923	1267.788462	8.865(
	SmokingJoes	35.217391	542.391304	7.257(
	Target	6.915254	1466.101695	11.77:
	Wal-Mart	6.494279	1685.926773	14.50(
	Walgreens	9.461538	990.384615	6.922(
VODKA	CVS	7.721739	1491.304348	11.42(
	Caseys	14.080986	625.968310	7.075(
	Hy-Vee	12.123082	943.710514	8.867(
	Kum&Go	9.135697	691.381418	11.13(
	Other	13.287791	831.647570	8.567(
	QuikTrip	18.611684	400.171821	5.552:
	SamsClub	10.014894	1109.787234	10.75(
	SmokingJoes	17.761310	489.586583	5.376(
	Target	7.123762	1380.198020	14.13(
	Wal-Mart	8.823467	1320.639535	12.94(
	Walgreens	11.438596	1057.163743	10.95'
	CVS	9.380282	1086.267606	19.28'
	Caseys	10.756303	742.016807	15.65(